Kauno technologijos universitetas

Informatikos fakultetas

Dirbtiniai neuroniniai tinklai

P176B101 Intelektikos pagrindai

Trečias laboratorinis darbas

Autorius
Gustas Klevinskas

Akademinė grupė
IFF-8/7

Vadovas
Lekt. Dr. Germanas Budnikas

Kaunas, 2021

Turinys

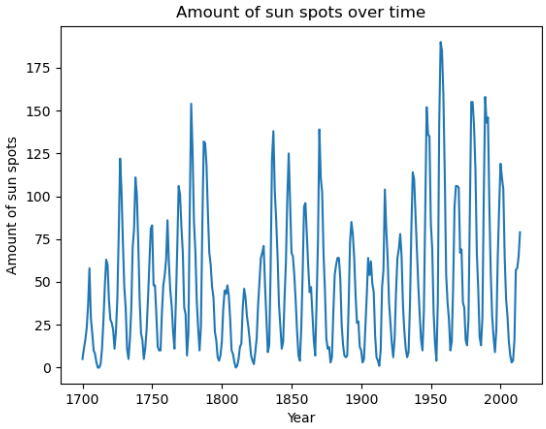
1. Įvadas

Laboratorinis darbas sudarytas iš dviejų dalių.

1. Susipažinti su prognozavimo uždavinio sprendimu panaudojant tiesinį dirbtinį neuroną, susipažinti su neuroninio tinklo mokymosi, testavimo ir jų panaudojimo uždaviniais.
2. Pritaikyti įgytas žinias kuriant modelį prognozavimo ar klasifikacijos uždaviniui spręsti naudojant pirmojo laboratorinio darbo duomenų rinkinį.

2. Pirma dalis

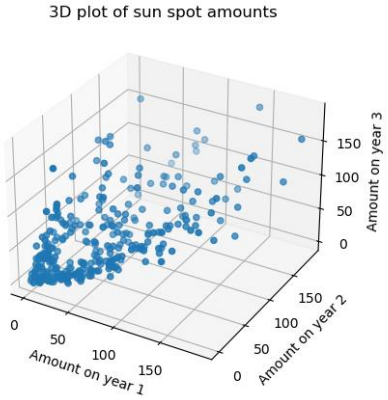
Pirmoji darbo dalis – panaudojant vienetinį dirbtinį neuroną sumodeliuoti saulės dėmių ateities reikšmę.



1 pav. *Ivesties duomenų grafikas*

1 pav. pateiktas grafikas, vaizduojantis saulės dėmių skaičių per metus. Iš šio grafiko matyti, jog duomenys yra užkraunami teisingai.

2.1. Modelis su eile 2



2 pav. *Trijų iš eilės einančių dėmių kiekio grafikas*

2 pav. pavaizduotos būsimos dvi įvesties vertės, ir norima trečioji vertė. Šios saulės dėmių kiekio vertės bus naudojamos kaip neurono įvestys ir norimos gauti vertės, skirtos neurono mokymui.

Atskyrimui sukurta funkcija, kurioje iteruojama per saulės dėmių kiekį ir atitinkamai priskiriamos P ir T matricų reikšmės. Ši funkcija pateikta kodo fragmente žemiau.

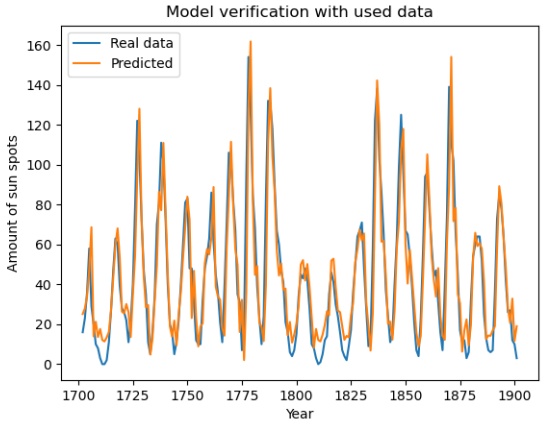
```
def get_input_output(spots):
    p = []
    t = []

    for i in range(len(spots) - N):
        p.append(spots[i: i + N])
        t.append(spots[i + N])

    return p, t
```

Išskirti pirmi 200 įverčių, kurie bus skirti neurono apmokymui. Šios reikšmės išskiriamos naudojant Python sąrašų *slice* funkciją. Išskirtos reikšmės perduodamos neurono apmokymo funkcijai, kuri randa optimalias svorių ir koeficiento *b* reikšmes. Šios dalies kodo fragmentas pateiktas žemiau.

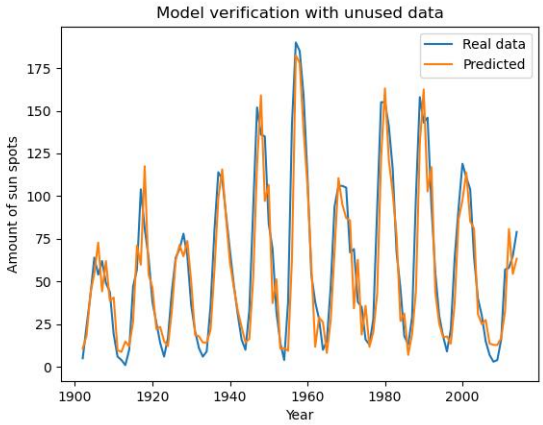
```
pu = p[:200]
tu = t[:200]
model = LinearRegression().fit(pu, tu)
```

Apmokius neuroną gautos šios svorių ir *b* reikšmės:

- $w_1 = -0.67608198$
- $w_2 = 1.37150939$
- $b = 13.403683236718116$



3 pav. *Tikri ir prognozuoti saulės dėmių kiekiai su mokymo duomenimis*

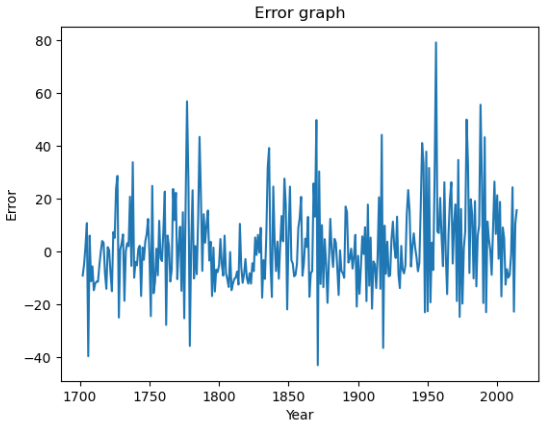


4 pav. *Tikri ir prognozuoti saulės dėmių kiekiai su nematytais duomenimis*

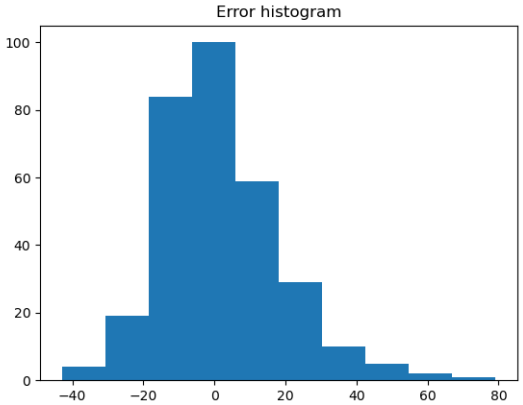
3 ir 4 pav. pateikti tikrų ir prognozuotų reikšmių grafikai. Pirmu atveju atlikta modelio verifikacija naudojant apmokymo duomenis, o antruoju – su modeliui dar nematytais duomenimis. Iš šių grafikų matyti,

jog neuronas apmokytas sėkmingai, grafiko forma gana artimai atitinka tikrąsias reikšmes. Šios dalies ir naudojamos pagalbinės grafiko spausdinimo funkcijos kodo fragmentai pateikti žemiau.

```
def verify(t, prediction, years, title):
    fig, ax = plt.subplots()
    real_data, = ax.plot(years, t)
    predicted, = ax.plot(years, prediction)

    ax.set_title(title)
    ax.set_xlabel('Year')
    ax.set_ylabel('Amount of sun spots')
    ax.legend([real_data, predicted], ['Real data', 'Predicted'])

tsu = model.predict(pu)
plots.verify(tu, tsu, years[N:200 + N], 'Model verification with used data')

tsu = model.predict(p[200:])
plots.verify(t[200:], tsu, years[200 + N:], 'Model verification with unused data')
```



5 pav. *Klaidos reikšmės*



6 pav. *Klaidos reikšmių histograma*

5 pav. pateikti klaidų dydžiai. Šios reikšmės parodo, kiek skyrėsi prognozuoti saulės dėmių kiekiai nuo tikrųjų. Tam buvo naudojami apmokymui skirti duomenys. Tiek iš šio, tiek iš 6 pav. pateiktos šio grafiko histogramos matyti, jog paklaida pagrinde buvo nedidelė – svyravo apie 20 dėmių kiekio nuo tikrojo. Vietomis matomos ir gana didelės paklaidos, tačiau histogramoje pateiktas jų dažnis yra mažas. Šių grafikų spausdinimui parašytas kodas pateiktas žemiau.

```
def error(e, years):
    fig, ax = plt.subplots()
    ax.plot(years, e)

    plt.title('Error graph')
```

```
    plt.xlabel('Year')
    plt.ylabel('Error')


def error_hist(e):
    fig, ax = plt.subplots()
    ax.hist(e)

    plt.title('Error histogram')

error = np.subtract(t, model.predict(p))
plots.error(error, years[N:])
plots.error_hist(error)
```

Papildomam modelio įvertinimui apskaičiuotos *MSE* ir *MAD* reikšmės. Jos suapvalintos iki dviejų skaičių po kablelio dėl tvarkos.

- $MSE = 278.27$
- $MAD = 9.22$

Šių dydžių skirtumus galima paaiškinti dėl jų skirtingų savybių. *MSE* yra jautri nuokrypiams, o *MAD* reikšmei nuokrypiai nedaro didelės įtakos, nes apskaičiavimui naudojamas vidurkis. Iš 6 pav. histogramos matyti, jog nuokrypiai yra nedažni palyginus su kitais dažniais, tad *MAD* klaidos įvertis yra mažesnis. Šių įverčių apskaičiavimui sukurtos funkcijos, kurios pateiktos žemiau.

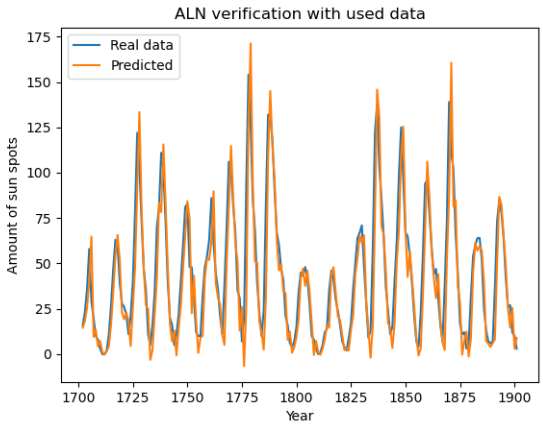
```
def mse(errors):
    return sum(map(lambda x: x ** 2, errors)) / len(errors)


def mad(errors):
    return np.median(np.abs(errors))
```

Sekančioje dalyje skaičiavimams naudojamas tiesinis neuronas. Jo sukūrimui panaudotas užduoties aprašyme pateiktas mokymosi šaltinis. Mokymui naudotos šios neurono reikšmės:

- Mokymosi greitis – 0.00000016
- Maksimalus epochų kiekis – 1000
- Siekiama mokymosi klaidos (*MSE*) reikšmė – 200

Optimalus mokymosi greitis rastas eksperimentiškai. Pradėta nuo 0.1, ir vis mažinta kol modelis pradėjo konverguoti. Su didesnėmis reikšmėmis iteruojant mokymosi procese gaunami milžiniški skaičiai, kurie perpildo atminį. Modelis gali tapti nekonverguojantis, jei naudojama per didelė mokymosi greičio reikšmė. Tuomet „prašokami“ optimalūs svorių sprendiniai ir jie vis didėja, kas priveda prie kintamųjų perpildymo. Modelio apmokymui ir verifikacijai naudotas kodas pateiktas žemiau.

```
aln = AdaptiveLinearNeuron(LEARNING_RATE, EPOCH_COUNT, ERROR_GOAL).fit(np.array(p[:200]), np.array(t[:200]))
prediction = aln.predict(np.array(p[:200]))
plots.verify(tu, prediction, years[N:200 + N], 'ALN verification with used data')
```



7 pav. *Tiesinio neurono rezultatai*

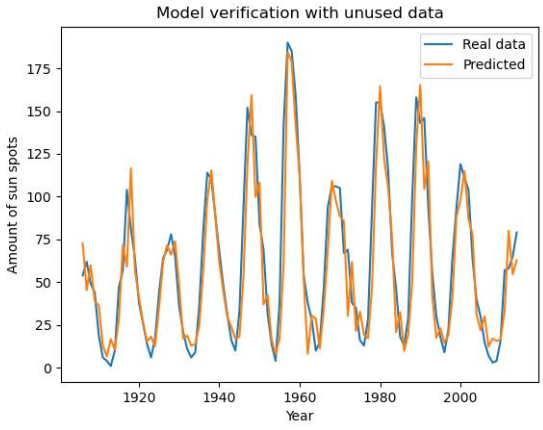
7 pav. pateiktas grafikas, vaizduojantis modelio prognozuotas reikšmes ir tikras saulės dėmių kiekių reikšmes. Matoma, jog prognozuojamos reikšmės artimos tikroms. Šio modelio *MSE* ir *MAD* įverčiai:

- $MSE = 268.39$
- $MAD = 8.07$

Šios reikšmės yra mažesnės nei pirmajame modelyje, tad galima teigti, jog šis apmokytas tiesinio neurono modelis geriau prognozuoja saulės dėmių kiekius.

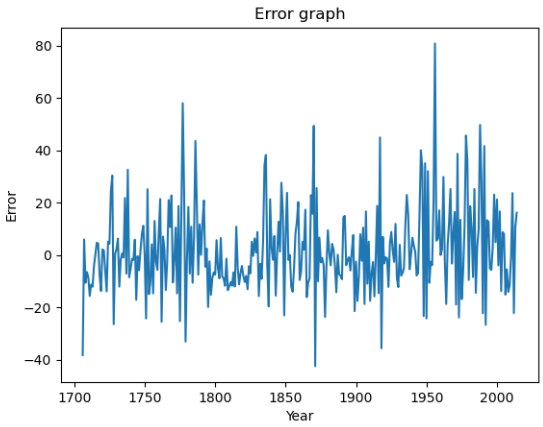
2.2. Modelis su eile 6

Programa sukurta taip, kad būtų galima lengvai keisti neurono eilę. Tam yra išskirtas globalus kintamasis programos viršuje – N, kuris nurodo modelio eilę. Šią reikšmę padidinus iki 6, gaunamos 8 pav. pavaizduotos prognozės.

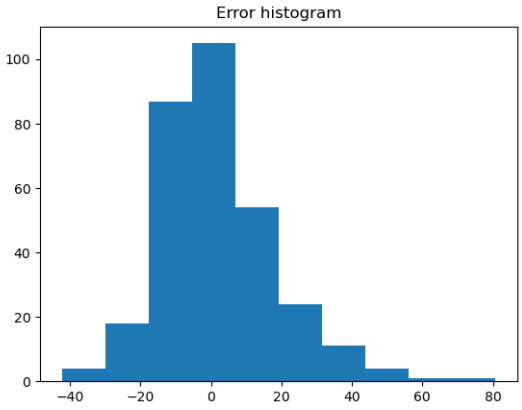


8 pav. *Modelio su 6 eile prognozės*

Iš grafiko matyti, jog jos, taip pat kaip ir su 2 eile, nedaug skiriasi nuo tikrųjų reikšmių. Papildomam modelio įvertinimui 9 – 10 pav. pateikti klaidų grafikai.



9 pav. *Modelio su 6 eile klaidos*



10 pav. *Modelio su 6 eile klaidų histograma*

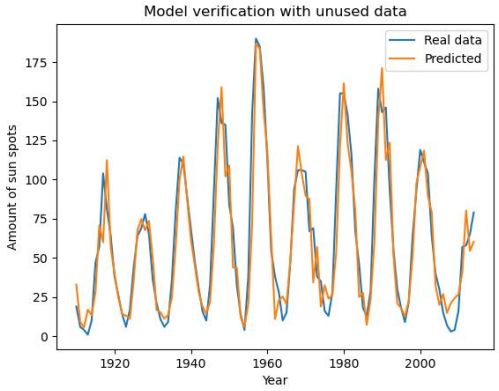
MSE ir *MAD* įverčiai šiam modeliui:

- $MSE = 271.04$
- $MAD = 8.96$

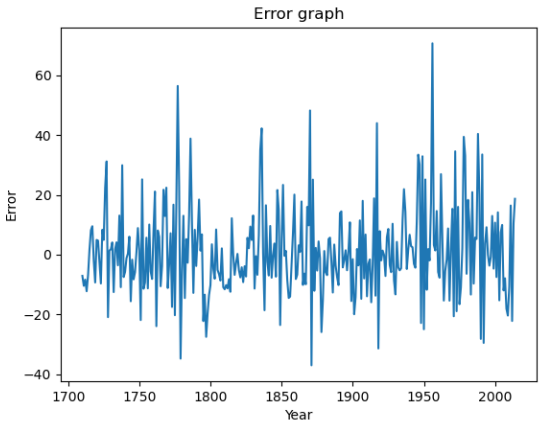
Palyginant su rezultatais, kai modelio eilė buvo 2, šios reikšmės yra mažesnės, bet skiriasi nedaug.

2.3. Modelis su eile 10

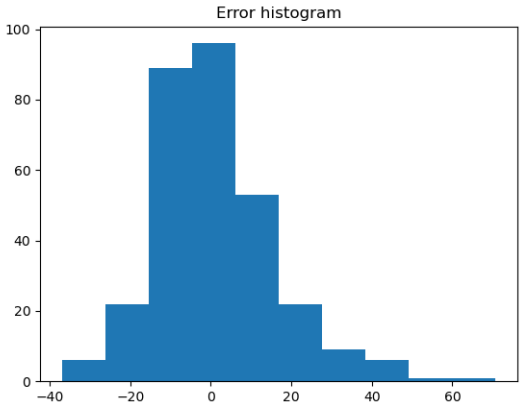
Eilę padidinus iki 10 gaunami žemiau pateikti grafikai.



11 pav. *Modelio su eile 10 prognozės*



12 pav. *Modelio su eile 10 klaidos*

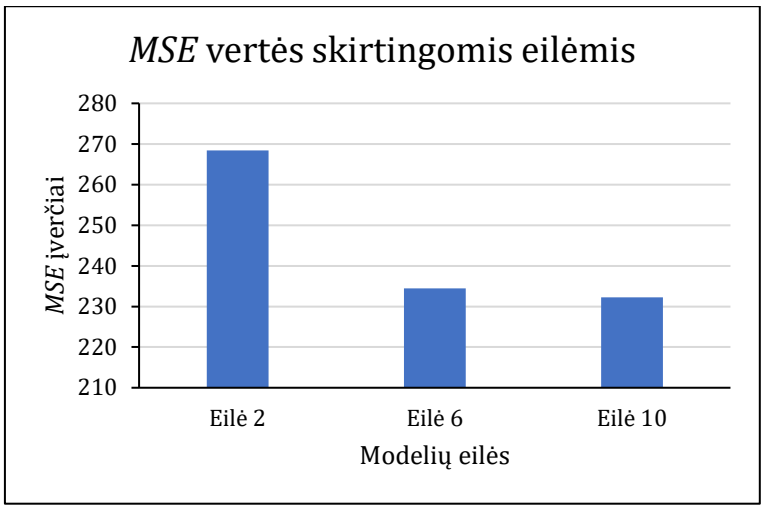


13 pav. *Modelio su eile 10 klaidų histograma*

Kaip ir praeituose modeliuose, šis modelis apmokytas sėkmingai – prognozuojamos reikšmės, pateiktos 11 pav., artimos tikrosioms. Klaidų grafikai, pateikti 12 ir 13 pav., beveik nesiskiria nuo anksčiau pateiktų modelių grafikų. Šio modelio *MSE* ir *MAD* įverčiai:

- $MSE = 232.26$
- $MAD = 8.61$

MAD įvertis skiriasi per kelias dešimtąsias nuo praeitų modelių, tačiau *MSE* įvertis ženkliai mažesnis palyginus su modeliu, kai eilė 2. Tai byloja, kad modelio prognozės, palyginus su ankstesniais žemesnių eilių modeliais, žymiai mažiau nukrypsta.



14 pav. *Modelių MSE įverčiai*

Šiam palyginimui pavaizduoti pateiktas 14 pav. grafikas. *MSE* įverčių ašis prasideda ne nuo nulio, kad būtų aiškiau matomi skirtumai.

3. Antra dalis

Pirmajame laboratoriniame darbe analizuota kompiuterinio žaidimo „League of Legends“ pirmųjų 10 minučių žaidimo statistika.

1 lentelė. *Tolydinių atributų charakteristikos*

Atributo pavadinimas	Kiekis	Trūkstamos reikšmės, %	Kardinalumas	Minimali reikšmė	Maksimali reikšmė	1-asis kvartilis	3-asis kvartilis	Vidurkis	Mediana	Standartinis nuokrypis
Blue kills	9879	0	21	0	22	4	8	6.184	8	3.011
Blue deaths	9875	0	21	0	22	4	8	6.137	4	2.934
Blue assists	9879	0	30	0	29	4	9	6.645	10	4.065
Blue total gold	9879	0	4739	10730	23701	15415	17459	16503.46	17828	1535.447
Blue total minions killed	9879	0	148	90	283	202	232	216.7	229	21.858
Red kills	9879	0	21	0	22	4	8	6.138	6	2.934
Red deaths	9879	0	21	0	22	4	8	6.184	8	3.011
Red assists	9879	0	28	0	28	4	9	6.662	3	4.061
Red total gold	9879	0	4732	11212	22732	15427	17419	16489.04	16786	1490.888
Red total minions killed	9879	0	153	107	289	203	233	217.349	231	21.912

2 lentelė. *Kategorinių atributų charakteristikos*

Atributo pavadinimas	Kiekis	Trūkstamos reikšmės, %	Kardinalumas	Moda	Modos dažnumas	Moda, %	2-oji moda	2-osios modos dažnumas	2-oji moda, %
Won	9876	0	2	red	4946	50.1	blue	4930	49.9
First blood	9879	0	2	blue	4987	50.5	red	4892	49.5
Blue dragons	9879	0	2	0	6303	63.8	1	3576	36.2
Blue towers destroyed	9879	0	5	0	9415	95.3	1	429	4.3
Red dragons	9879	0	2	0	5798	58.7	1	4081	41.3
Red towers destroyed	9879	0	3	0	9483	96	1	367	3.7

Pasirinkta prognozuoti, ar mėlynoji komanda laimėjo žaidimą atsižvelgiant į šios komandos atliktų nužudymų, mirčių, padėjimų kiekius ir bokštų nugriovimus. Didžioji dalis originalių duomenų, susijusių su raudonąja komanda bei kitais nereikalingais įverčiais (pirmąjį nužudymą atlikusi komanda, sukauptų pinigų kiekis, monstrų nugalabijimai), buvo pašalinti, nes jie nebuvo naudojami kuriant modelį.

Stulpelių ištrynimas įvykdytas kitomis programomis, tačiau paliktas „Minions killed“ stulpelis testavimo tikslams. Pamačius, kad jis nedaro ženklios įtakos prognozavimui, jis programiškai ištrintas. Kategorinės reikšmės „Won“ duomenys pakeisti iš komandos spalvos į 0 – laimėjo raudonoji komanda, ir 1 – laimėjo mėlynoji komanda.

Darbui su failu naudota *pandas* biblioteka. Failo nuskaitymo kodo fragmentas pateiktas žemiau.

```
data = pd.read_csv(INPUT_FILE) \
    .head(INPUT_LIMIT) \
    .replace({'Won': {'blue': 1, 'red': 0}}) \
    .drop(columns=['Minions killed'])
```

Modelio bei įvesties ir išvesties duomenų sukūrimo kodo fragmentas pateiktas žemiau.

```
x = data.iloc[:, 1:].to_numpy()
y = data.iloc[:, 0].to_numpy()

model = Sequential()
model.add(Dense(1, input_dim=x.shape[1]))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer=Adam(learning_rate=LEARNING_RATE), loss='binary_crossentropy', metrics=['accuracy'])
```

Modelio kūrimui naudota *keras* biblioteka. Įvesties neuronui naudota numatyta aktyvacijos funkcija – tiesinė, o išvesties – sigmoidinė, siekiant apriboti reikšmes tarp 0 ir 1. Modelio prognozes tuomet bus interpretuojamos kaip modelio užtikrintumas, kad laimėjo mėlynoji komanda.

Parinktos pradinės modelio reikšmės:

- Mokymosi greitis – 0.5
- Maksimalus epochų kiekis – 15

Parinktas mažas epochų kiekis, nes su didesniais tinklo tikslumas negerėdavo, o mažesnė vertė padidina greitaveiką. Dėl panašios priežasties apribotas ir duomenų kiekis iki 700 eilučių.

Kryžminės patikros rezultatai pateikti 3 lentelėje.

3 lentelė. *Kryžminės patikros rezultatai*

Eil. nr.	Nuokrypis	Tikslumas
1.	0.51	70.00%
2.	0.72	55.71%
3.	0.87	67.14%
4.	0.89	72.86%
5.	0.88	30.00%
6.	0.41	85.71%
7.	0.54	74.29%
8.	0.75	65.71%
9.	0.58	67.14%
10.	0.76	57.14%

Vidutinės reikšmės:

- Nuokrypio – 0.69

- Tikslumo – 64.57%

Kryžminės patikros kodo fragmentas pateiktas žemiau.

```
fold_num = 1
test_loss = 0.0
test_acc = 0.0

for train, test in KFold(n_splits=10, shuffle=False).split(x, y):
    model.fit(x[train], y[train], batch_size=5, epochs=15, verbose=0)
    scores = model.evaluate(x[test], y[test], verbose=0)
    print('{:>2}. Score: {} of {:.2f}; {} of {:.2f}%'.format(
        fold_num, model.metrics_names[0], scores[0], model.metrics_names[1], scores[1] * 100))

    test_loss += scores[0]
    test_acc += scores[1]
    fold_num += 1

print('\n--- Results ---')
print('Average loss: {:.2f}'.format(test_loss / (fold_num - 1)))
print('Average accuracy: {:.2f}%'.format(test_acc * 100 / (fold_num - 1)))
```

Modelio patikrinimui sugeneruotos prognozės ir 4 lentelėje pateikti rezultatai. Įvesties stulpeliai: „Kills“, „Deaths“, „Assists“ ir „Dragons“, o išvesčiai naudota „Won“. Modelio prognozė pateikta stulpelyje „Predicted won“.

4 lentelė. *Duomenų fragmentas su prognozėmis*

Row	Won	Predicted won	Kills	Deaths	Assists	Dragons
500	1	0.938373	10	6	11	1
501	1	0.780010	6	4	9	1
502	1	0.938905	13	6	25	0
503	0	0.059286	2	9	4	0
504	0	0.334860	4	5	4	0
505	0	0.104363	2	7	2	0
506	1	0.957992	9	3	8	1
507	0	0.078711	3	10	3	0
508	1	0.909017	8	4	9	1
509	1	0.847280	6	3	5	1

Iš duomenų fragmento matyti, jog modelis gana gerai prognozuoja būsimą žaidimo laimėtoją. Siekiant pagerinti modelį, buvo pakeista modelio vertė:

- Mokymosi greitis – 0.1

Su šia nauja verte taip pat atlikta kryžminė patikra. Jos rezultatai pateikti 5 lentelėje.

5 lentelė. *Kryžminės patikros rezultatai*

Eil. nr.	Nuokrypis	Tikslumas
1.	0.44	80.00%
2.	0.59	70.00%
3.	0.56	75.71%
4.	0.51	78.57%
5.	0.60	72.86%
6.	0.46	85.71%

7.	0.49	77.14%
8.	0.49	80.00%
9.	0.57	72.86%
10.	0.70	61.43%

Vidutinės reikšmės:

- Nuokrypio – 0.54
- Tikslumo – 75.43%

Su nauja mokymosi greičio verte, vidutinis modelio tikslumas pagerintas 11%. 6 lentelėje pateikti šio modelio gautos prognozės tam pačiam duomenų segmentui.

6 lentelė. *Duomenų fragmentas su prognozėmis*

Row	Won	Predicted won	Kills	Deaths	Assists	Dragons
500	1	0.930504	10	6	11	1
501	1	0.812955	6	4	9	1
502	1	0.953236	13	6	25	0
503	0	0.036791	2	9	4	0
504	0	0.273888	4	5	4	0
505	0	0.069319	2	7	2	0
506	1	0.957992	9	3	8	1
507	0	0.039939	3	10	3	0
508	1	0.914183	8	4	9	1
509	1	0.853187	6	3	5	1

Matyti, jog visos prognozės pagerėjo. Kai komanda laimi, prognozė artimesnė vienetui nei praeito modelio prognozė, ir atitinkamai, kai komanda pralaimi – prognozė artimesnė nuliui.

Tačiau 504 eilutės prognozė išliko palyginus aukšta, nors komanda pralaimėjo. Tai galima paaiškinti tuo, jog pirmosios 10 žaidimo minučių nėra lemiamos. Per likusį žaidimo laiką laimėjimo persvara gali pereiti kitai komandai, o šis duomenų rinkinys to neapima. Tikėtina, jog ši 10 minučių riba parinka, kad visi duomenys būtų lygiaverčiai – kuo ilgesnis žaidimas, tuo didesni nugalabijimų, mirčių, pinigų kiekiai.

4. Išvados

Pirmosios dalies išvados:

1. Reikia palikti dalį duomenų rinkinio modelio verifikacijai, kad būtų patikrintas modelis su nematytais duomenimis.
2. *MSE* reikšmė jautri nuokrypiams, o *MAD* reikšmė – tam atspari.
3. Didinant modelio eilę gaunamos vis tikslesnės prognozės. Tačiau tai kainuoja papildomus apskaičiavimus ir kartu greitaveikos.

Antrosios dalies išvados:

1. Įvesčių neuronų skaičius ir duomenų kiekis turi sutapti.
2. Modelio tikslumas stipriai priklauso nuo jam paduodamų duomenų kokybės ir susiejimo su išvestimi. Antroje dalyje naudoto žaidimo „League of Legends“ laimėjimo prognozė tik dalinai atitinka to duomenų rinkinio duomenis.
3. Kryžminė validacija leidžia efektyviai patikrinti modelio veikimą.